الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE DELPHI

SÉRIE 05

LEÇON 1 : AJOUT DES FICHES

OBJECTIF PÉDAGOGIQUE :

À l'issue de cette leçon, les stagiaires seront capables d'inculquer les notions d'un fichier et l'appliquer sur un programme.

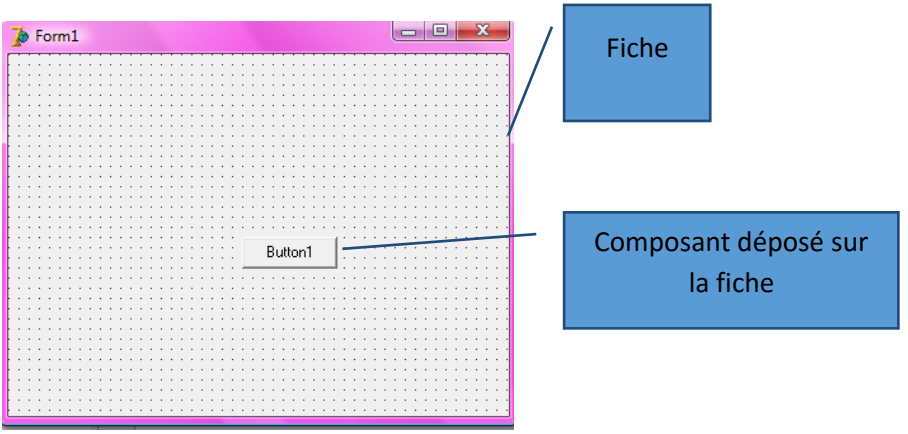
PLAN DE LA LEÇON

I- **LE RÔLE D'UNE FICHE**

II- **AJOUTER UNE FICHE DANS UNE APPLICATION**

III- **FAIRE APPEL AUX FICHES**

I- LE RÔLE D'UNE FICHE :



La fiche est une fenêtre sur laquelle on peut déposer des objets (composants) qui vont constituer les éléments de l'interface utilisateur.

II-AJOUTER UNE FICHE DANS UNE APPLICATION :

Afin d'ajouter une fiche dans une application, il suffit de cliquer sur **Fichier > Nouvelle fiche :** cette commande vous permet de rajouter d'autres fiches à votre application selon le besoin, la fiche insérée sera superposée à celle déjà existantes.

III- FAIRE APPEL AUX FICHES :

Pour appeler une fiche, il existe deux méthodes **:**

- Form1.Show
- Form1.ShowModal

Show/ShowModal Il y a deux différences majeures :
1/

- Form1.Show ne bloque pas le programme, donc toutes les instructions situées après le Show sont exécutées dès que la fiche est affichée.
- Form1.ShowModal bloque le programme, les instructions suivantes ne seront exécutées qu'à la fermeture de la fiche.

2/

Une fenêtre affichée avec ShowModal bloque l'accès à toutes les autres fenêtres tant que celle-ci n'est pas fermée. Avec le Show ce n'est pas le cas.

Exemple :

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Form2.ShowModal;
ShowMessage('SHOWMODAL:Ce message s''affiche une fois que Form2 est fermée');
end;
```

Dans ce cas tant que Form2 est visible il n'est pas possible de cliquer sur Form1 même en déplaçant Form2. C'est le cas typique des boites de dialogue qui attendent des saisies avant de continuer.

```
procédure TForm1.Button1Click(Sender: TObject);
begin
Form2.Show;
ShowMessage('SHOW:Ce message s''affiche de suite');
end;
```

Dans ce cas il est possible d'accéder à Form1 sans fermer Form2. C'est le cas principal des fenêtres d'outils ou de volet de visualisation. Les fiches affichées comme ceci ont souvent la propriété StayOnTop de manière à rester visible (mais ce n'est pas obligatoire).

Quant à la dernière méthode c’est celle qui permet de créer des fiches à l'exécution seulement quand on en a besoin. Par défaut toutes les fiches de Delphi sont créées automatiquement au démarrage. On peut choisir dans les options du projet celles qui ne seront pas créés automatiquement. Mais attention : si une fiche n'est pas créée, aucun des composants qu'elle contient n'est accessible. De même après le Release tous les composants de la fiche sont détruits. Il faut donc sauver avant les valeurs utiles (entre le ShowModal et le Release). L'utilisation de cette méthode est surtout réservée aux fiches servant ensuite de composants comme les OpenDialog ou SaveDialog. Dans la plupart des applications cette méthode est inutile.

LEÇON 2 : ETUDE DES COMPOSANTS LES PLUS UTILISÉES

OBJECTIF PÉDAGOGIQUE : À l'issue de cette leçon ; les stagiaires seront capables de utilisées des composants dans le programme de Delphi **7.**

PLAN DE LA LEÇON :

I- BOUTONS

1. Le composant Button
2. Le composant Bit BTN
3. Le composant Speed Button

II- BARRE D'ÉTAT STATUS BAR

III- BARRE D'ÉDITION

1. Le composant Edit
2. Le composant Mask Edit
3. Fonction de conversion

   StrToInt

   IntToStr
4. Fonctions Lenght et compare texte Lenght

IV- LES PANNEAUX (PANEL)

V- CASE À COCHER (CHECK BOX)

VI-CASE D'OPTIONS (RADIO BUTTON, GROUPE BOX)

VII-BOITE DE LISTE (LIST BOX)

VIII- BOITE COMBO (COMBO BOX)

IX-LES ZONES D'AFFICHAGE (LABEL, BEVEL)

X-TIMER

APPLICATION

LEÇON 2 : ETUDE DES COMPOSANTS LES PLUS UTILISÉES

I. BOUTONS :

1. Le composant Button :

Ce composant sert en général à proposer à l'utilisateur une action. Cette action lui est expliquée par un texte très court sur le bouton, du style "OK" ou "Annuler". Lorsque l'utilisateur clique sur le bouton, ou appuie sur Espace ou Entree lorsque celui-ci est sélectionné (ce qui revient à le cliquer), l'événement OnClick se produit, qui offre une possibilité de réaction. C'est en général dans la procédure de réponse à cet événement qu'on effectue l'action proposée à l'utilisateur, comme afficher ou fermer une fiche par exemple pour citer des exemples récents.

Les boutons ont une fonctionnalité en rapport avec les fenêtres modales : ils ont une propriété ModalResult. Cette propriété ne fonctionne pas du tout comme celle des fiches. Elle est constante (fixée par vous), et est recopiée dans la propriété ModalResult de la fiche lors d'un clic sur le bouton. Ceci a comme effet de pouvoir créer un bouton "OK" rapidement en lui donnant "mrOK" comme ModalResult. Lors d'un clic sur ce bouton, la propriété ModalResult de la fiche devient mrOK et la fiche se ferme donc, sans aucune ligne de code source écrite par nous.

Ce composant a certaine propriété spécifique.

Propriété	Description
Cancel	Détermine si le gestionnaire d'événement OnClick du bouton doit s'exécuter quand l'utilisateur appuie sur la touche Echap.
Default	Détermine si le gestionnaire d'événement OnClick du bouton doit s'exécuter quand l'utilisateur appuie sur la touche Entrée.
ModalResult	Détermine si le choix de ce bouton ferme sa fiche parent (modale) et comment cette fermeture a lieu.

2. Le composant Bit BTN :

C'est un contrôle bouton poussoir qui peut contenir une image. Ce composant a également certaine propriété spécifique.

Propriété	Description
Glyph	Détemine le bitmap qui sera dans le bouton.
Kind	Détermine le type du bouton.
Layout	Détermine l'emplacement de l'image dans un bouton bitmap.
Margin	Détermine le nombre de pixels entre le côté de l'image et le côté du bouton.
NumGlyphs	Permet d'indiquer le nombre d'image dans le bitmap.
Spacing	Détermine le nombre de pixels entre l'image et le texte.
Style	Détermine l'aspect du bouton.

3. Le composant speed button :

SpeedButton est utilisé pour ajouter un bouton à un groupe de boutons dans une fiche.

Ce composant a certaine propriété spécifique.

Propriété	Description
AllowAllUp	Détermine si tous les boutons d'un groupe peuvent être simultanément non sélectionnés.
Down	Permet au bouton de rester sélectionner (enfoncé).
Flat	Détermine si le bouton a une bordure 3D.
Glyph	Détermine le bitmap qui sera dans le bouton.
GroupIndex	Permet à un certain nombre de bouton de travailler en groupe.
Layout	Détermine l'emplacement de l'image dans un bouton bitmap.
Margin	Détermine le nombre de pixels entre le côté de l'image et le côté du bouton.
NumGlyphs	Permet d'indiquer le nombre d'image dans le bitmap.
Spacing	Détermine le nombre de pixels entre l'image et le texte.
Transparent	Détermine si l'arrière-plan du bouton est transparent.

II. BARRE D'ÉTAT STATUS BAR

Comme dans toute application Windows, la barre d'outils permet l'accès rapide à certaines commandes des menus. Il est impossible ici de décrire chaque bouton étant donné qu'ils changent à chaque version de Delphi. Pour avoir une description de chaque bouton, lisez la bulle d'aide qui ne manquera pas de s'afficher lors du passage de la souris.

La barre d'outils pourra, dès la version 2, être personnalisée pour être agrémentée à vos goûts et habitudes. Pour cela, effectuez un clic droit sur l'une des barres et une commande « personnaliser» devrait être disponible. Les manipulations pour personnaliser les barres sont similaires à celles qu'on rencontre dans des logiciels tels Microsoft Word.

III. BARRE D'ÉDITION :

1. Le composant Edit :

Les composants "Edit" permettent de proposer des zones d'édition. Ces zones très souvent utilisées sous Windows ne contiennent qu'une ligne de texte, dont la police peut être réglée, toujours avec la même parcimonie. Ce composant permet à l'utilisateur d'entrer une information quelconque tapée au clavier. Le texte entré dans le composant est accessible via une propriété "Text". Il est possible de fixer une limite à la longueur du texte entré, de masquer les caractères (utile pour les mots de passe), de désactiver la zone ou d'interdire toute modification du texte.

Propriétés :

AutoSelect	Permet l'autosélection du contenu lors de l'activation : lorsque le contrôle devient actif, le texte est sélectionné, de sorte qu'il peut directement être modifié en tapant un nouveau texte. Utiliser cette fonction dans les formulaires peut faire gagner un temps précieux.
Enabled	Active ou désactive la zone d'édition (l'édition n'est possible que lorsque la zone est activée).
MaxLength	Permet de fixer le nombre maximal de caractères entrés dans la zone. Mettre 0 pour ne pas donner de limite (par défaut).
PasswordChar	À utiliser lorsqu'on veut masquer les caractères tapés, comme pour les mots de passe. Utiliser le caractère "*" pour masquer (le plus souvent utilisé) et "#0" (caractère n°0) pour ne pas masquer.
ReadOnly	Permet d'activer la lecture seule de la zone d'édition. Lorsque "ReadOnly" vaut "true", la lecture est toujours possible mais l'écriture impossible.

Text	Contient le texte entré dans la zone d'édition. C'est aussi en changeant cette propriété que l'on fixe le contenu de la zone.

2. Le composant Mask Edit :

L'utilisation de maskedit est très simple. D'abord son role est de vérifier la saisie des utilisateurs.
En modifiant la propriété "editmask" on pourra définir les mask.

Aussi il y a la propriété "text" qui réceptionne la valeur du editmask .

3. Fonction de conversion :

StrToInt

```
function StrToInt(Value:string):integer;
```

Permet de convertir une chaîne de caractères en son nombre entier écrit dedans.

Exemple

```
uses SysUtils, Dialogs;
begin
  if StrToInt('50')=50 then
    ShowMessage('C''est vrai');
end.
```

IntToStr

```
function IntToStr(Value:integer):string;
```

Permet d'afficher un nombre entier dans une chaîne de caractères.

Exemple

```
uses SysUtils, Dialogs;
begin
```

```
  ShowMessage(IntToStr(50));
end.
```

4. Fonctions Lenght et compare texte :

Lenght

```
function Length(Value:string):integer;
```

Renvoie la longueur de la chaîne.

Exemple

```
uses SysUtils, Dialogs;
begin
ShowMessage(IntToStr(Length('Vaut 6')));
end.
```

Compare texte

Permet de comparer deux chaines de caractère.

IV. LES PANNEAUX (PANEL) :

Les composants "Panel", ou panneaux, sont fréquemment employés sous Delphi. Ils ont deux rôles connus, que l'on peut parfois combiner. L'intérêt de ces panneaux, c'est que ce sont des conteneurs, qu'ils possèdent une bordure 3D configurable à volonté, et qu'ils sont alignables. Cette dernière caractéristique, si elle n'évoque probablement pas grand-chose pour vous, sera bientôt une aide précieuse dans la conception de vos interfaces. Les panneaux sont donc utilisés soit pour leurs bordures 3D, soit pour créer des alignements.

Au niveau du code, les panneaux sont assez peu manipulés, sauf lorsqu'on a besoin de règler leurs dimensions pour maintenir l'interface d'une application. A la conception, par contre, il est possible de manipuler quelques propriétés agissant sur l'alignement et l'apparence des panneaux.

Propriétés :

Align	Décide de l'alignement du panneau **:** • alNone : aucun alignement. • alLeft : alignement sur la gauche du conteneur. Seule la bordure droite du panneau est libre. • alRight : alignement sur la droite du conteneur. Seule la bordure gauche du panneau est libre. • alTop : alignement en haut du conteneur. Seule la bordure inférieure du panneau est libre. • alBottom : alignement en haut du conteneur. Seule la bordure supérieure du panneau est libre. • alClient : alignement sur toute la partie non recouverte par d'autres composants alignés. Aucune bordure n'est libre.
BevelInner	Type de bordure intérieure. Vous avez le choix entre bvLowered (effet abaissé), bvNone (pas d'effet), bvRaised (effet relevé) et bvSpace (espacement). Essayez ces effets pour voir ce qu'ils donnent concrètement.
BevelOuter	Item avec la bordure extérieure. Combinez les deux bordures pour obtenir un cadre à effet relevé ou enfoncé.
BevelWidth	Largeur des bordures intérieures et extérieures en pixels. Evitez de fixer une valeur supérieure à 3 pixels.
BorderWidth	Spécifie l'espacement entre la bordure intérieure et la bordure extérieure. Essayez de ne pas dépasser quelques pixels.
Caption	Spécifie le texte qui apparaît à l'intérieur du panneau. Lorsque le panneau est utilisé pour créer des alignements, il faut penser à effacer le contenu de Caption.

V. CASE À COCHER (CHECK BOX) :

Un composant CheckBox permet de donner à l'utilisateur un choix de type "Oui/Non". S'il coche la case, la réponse est "Oui", sinon, c'est "Non". Au niveau du code, une propriété de type booléen stocke cette réponse : Checked. Il est possible de donner un texte explicatif de la case à cocher dans la propriété Caption. Ce texte apparait à côté de la case. Une modification de l'état de la case déclenche un événement OnClick, que cette modification provienne effectivement d'un clic ou d'une pression sur la touche Espace. La modification depuis le code source de la propriété Checked ne déclenche pas cet événement.

Propriétés :

Caption	Permet de spécifier le texte qui apparait à côté de la case à cocher. Il est à noter que le composant ne se redimensionne pas pour afficher automatiquement tout le texte. Ce sera donc à vous de prévoir cela si c'est nécessaire.
Checked	Permet de connaître ou de modifier l'état de la case : cochée ou pas.

Evénements **:**

OnClick	Déclenché lorsque l'état de la case à cocher change, quelle que soit l'origine du changement. La propriété Checked est déjà mise à jour lorsque la procédure de réponse à cet événement s'exécute.

VI. CASE D'OPTIONS (RADIO BUTTON, GROUPE BOX) :

- Le composant "RadioButton" permet de proposer un choix à l'utilisateur parmi plusieurs possibilités. Chaque possibilité est constituée d'un composant "RadioButton". A l'intérieur d'un composant conteneur ou d'une fiche, l'utilisateur ne peut cocher qu'un seul composant "RadioButton", quel que soit le nombre proposé. Pour créer des groupes de composants séparés, il faut les regrouper dans des contrôles conteneurs comme les "Panel" ou les "GroupBox". Pour savoir quel RadioButton est coché, il faut passer en revue leur propriété Checked.

En ce qui concerne les propriétés et événemnts, un "RadioButton" est assez proche d'un "CheckBox". Voici, comme d'habitude, un résumé des éléments à connaître.

Propriétés :

Caption	Permet de spécifier le texte qui apparait à côté de la case d'option. Il est à noter que le composant ne se redimensionne pas pour afficher automatiquement tout le texte. Ce sera donc à vous de prévoir cela si c'est nécessaire.
Checked	Permet de connaître ou de modifier l'état de la case : cochée ou pas.

Evénements :

OnClick	Déclenché lorsque l'état de la case d'option change, quelle que soit l'origine du changement. La propriété Checked est déjà mise à jour lorsque la procédure de réponse à cet événement s'exécute.

Les composant "GroupBox" ont pour but d'effectuer des regroupements visibles de composants. En effet, ils se présentent sous la forme d'un cadre 3D et d'un texte en haut à gauche. Un composant GroupBox est un conteneur, donc susceptible de contenir d'autres composants (conteneurs ou pas, soit dit en passant). Lorsque l'on veut proposer à l'utilisateur le choix parmi plusieurs options, un composant GroupBox contenant des composants RadioButton est un bon choix. Vous trouverez facilement des exemples en vous promenant dans l'interface de vos logiciels favoris. Si vous comptez vous limiter à des RadioButton placés sur un GroupBox, intéressez-vous au composant RadioGroup non décrit ici mais qui permet de créer rapidement ce genre d'interface.

Au niveau du code, un composant GroupBox n'a pas souvent de rôle particulier à jouer, étant donné que c'est surtout son caractère conteneur qui importe. Aucune propriété, méthode ou événement n'est à signaler ici, cependant, il ne faut pas sous-estimer ce composant qui tient un grand rôle dans l'organisation tant visuelle que logique des interfaces les plus fournies.

VII. BOITE DE LISTE (LIST BOX) :

Ce composant permet d'afficher une liste d'éléments, parmi lesquels l'utilisateur peut choisir un ou plusieurs éléments (plusieurs choix sont possibles à ce niveau). Chaque élément est présent sur une ligne, un peu à la façon d'un mémo, mais il n'est pas question ici d'éditer les éléments, juste d'en sélectionner. Le composant ListBox n'est pas limité à proposer des éléments à l'utilisateur : il peut aussi servir à afficher une liste des résultats d'une recherche par exemple.

Au niveau du fonctionnement, la propriété objet Items permet la gestion des éléments. Cette propriété dispose d'une propriété tableau par défaut pour accèder aux éléments. Ces éléments sont de type chaîne de caractère. Items dispose en outre d'une méthode "Clear" pour vider la liste, d'une méthode "Add" pour ajouter un élément, et

d'une propriété "Count" pour indiquer le nombre d'élément. Les éléments sont numérotés de 0 à Count - 1. Outre la propriété Items, le composant ListBox dispose d'un certain nombre de propriétés permettant de configurer le composant.

Propriétés :

Extended Select	Lorsque MultiSelect vaut True, ExtendedSelect permet une sélection d'éléments non à la suite les uns des autres. Lorsque ExtendedSelect vaut False, seule une plage contigüe d'éléments peut être sélectionnée : d'un élément de départ à un élément de fin. Dans le cas contraire, n'importe quel élément peut être sélectionné ou non.
Integral Height	Permet ou interdit l'affichage d'éléments coupés, c'est-à-dire d'éléments partiellement visibles. Lorsque IntegralHeight vaut False, ces éléments sont affichés. Dans le cas contraire, la hauteur de la zone de liste est ajustée pour afficher un nombre entier d'éléments.
ItemHeight	Permet de fixer la hauteur d'un élément. Cette valeur est calculée automatiquement lorsque vous changez la police utilisée par la zone d'édition. Vous pouvez cependant spécifier une autre valeur, à vos risques et périls (!). Cette valeur est utilisée lorsque vous fixez IntegralHeight à True pour calculer la hauteur de la zone de liste.
MultiSelect	Permet de sélectionner plusieurs éléments. Lorsque MultiSelect est faux, un élément au plus est sélectionné. Dans le cas contraire, plusieurs éléments sont sélectionnables. Les possibilités de sélection dépendant de la valeur de ExtendedSelect.
Sorted	Lorsque Sorted vaut True, les éléments de la liste sont automatiquement triés et insérés par ordre alphabétique.

	Cette fonction est parfois très pratique bien que nous préférerons souvent utiliser nos propres tris ou ne pas trier (fixer Sorted à False).
Style	Fixe le style de la zone de liste. Pour l'instant, laissez la valeur de Style à lbStandard. Plus tard, si vous le souhaitez, vous pourrez en utilisant les autres valeurs dessiner vous-mêmes les éléments de la zone et créer ainsi des listes personnalisées visuellement.

Evénements :

OnClick	Se produit lorsqu'un clic se produit dans la liste.

VIII. BOITE COMBO (COMBO BOX) :

Le composant "ComboBox" (zone de liste déroulante) permet également la gestion de liste d'éléments, mais à d'autres fins et avec d'autres possibilités. En premier lieu, un composant "ComboBox" ne permet la sélection que d'un élément au plus. Le composant affiche une zone montrant selon les cas l'élément sélectionné ou une zone d'édition (analogue à un "Edit"). Il est possible d'afficher une liste des éléments de la liste en cliquant sur un bouton prévu à cet effet. Il est alors possible de sélectionner un élément dans la liste.

Au niveau du code, une composant ComboBox fonctionne comme un composant ListBox pour ce qui est de la propriété Items. Un composant ComboBox pouvant contenir une zone d'édition, il dispose de certaines éléments de ces dernières, comme les propriétés Length et Text, et l'événement OnChange. La propriété Style permet de choisir parmi 5 styles différents dont 3 seront décrits ici, les 2 autres autorisant la création d'éléments personnalisés visuellement, ce qui déborde largement du cadre de ce chapitre.

Propriétés :

MaxLength	Lorsque le composant comporte une zone d'édition, MaxLength fixe le nombre maximal de caractères entrés dans cette zone.
Sorted	Permet comme pour une ListBox d'activer le tri alphabétique des éléments.
Style	Détermine le style de la zone de liste déroulante. Voici la description de 3 des 5 valeurs proposées : • csDropDown : crée une liste déroulante avec une zone d'édition. Seule cette dernière est visible et permet d'entrer un élément qui n'est pas dans la liste. Le fait de taper du texte dans la zone d'édition déselectionne tout élément sélectionné.

OnChange	Lorsque le composant comporte une zone d'édition, celle-ci déclenche des événements OnChange comme une zone d'édition normale. Ces événements vous permettent d'exécuter du code répondant à tout changement.
OnClick	Déclenché lors d'un clic sur toute partie du composant, que ce soit la liste.
	• csDropDownList : crée une liste déroulante sans zone d'édition. Le composant montre l'élément sélectionné ou une zone vide si aucun ne l'est. Il faut obligatoirement choisir parmi les éléments proposés. • csSimple : crée une liste avec une zone d'édition. Les deux sont visibles. L'élément sélectionné est affiché dans la zone d'édition qui permet de taper un autre élément.
Text	Contient le texte de la zone d'édition. Attention, n'utilisez pas cette propriété si la propriété Style est fixée à csDropDownList car le texte est alors imprévisible (je dois avouer mon ignorance à ce sujet).

Evénement

XI. LES ZONES D'AFFICHAGE (LABEL, BEVEL) :

- Un composant "Label" permet d'inclure facilement du texte sur une fiche. Ce texte n'est pas éditable par l'utilisateur et c'est donc un composant que l'on utilise souvent comme étiquette pour d'autres contrôles. Bien que l'on puisse modifier la police utilisée pour l'écriture du texte, il faut toujours freiner ses ardeurs. Ces composants sont en général en grand nombre sur les fiches importantes, mais cela ne pose pas de problème car ils ne prennent presque pas de mémoire.Un composant "Label" peut contenir jusqu'à 255 caractères, ce qui le limite à des textes très courts. Les propriétés "AutoSize" et "WordWrap" permettent d'obtenir une bande de texte à largeur fixe sur plusieurs lignes, ce qui sert souvent pour donner des descriptions plus étoffées que de simples étiquettes. Les composants "Label" sont très souvent utilisés et le seront donc dans les manipulations futures.

Propriétés :

Alignment	Permet d'aligner le texte à droite, au centre ou à gauche. N'est utile que lorsque "AutoSize" est faux et qu'une taille différente de celle proposée a été choisie. Le texte s'aligne alors correctement.
AutoSize	Active ou désactive le redimensionnement automatique du Label. "true" ne permet qu'une seule ligne, avec une taille ajustée au texte présent dans le label, tandis que "false" permet plusieurs lignes, non ajustées au contenu du label.
Caption	Permet de spécifier le texte à afficher dans le label.
FocusControl	Référence à un autre contrôle. Cette propriété permet de choisir le composant qui reçoit le focus (qui est activé) lorsqu'on clique sur le label. Cette propriété permet un certain confort à l'utilisateur puisqu'un

clic sur une étiquette pourra par exemple activer le composant étiquetté.

Layout — Alter-ego vertical de "Alignment".

Visible — Permet de montrer ou de cacher le label. Cette propriété fait partie des grands classiques qui ne seront plus repris dans la suite.

WordWrap — Autorise les retours à la ligne pour permettre d'afficher plusieurs lignes à l'intérieur du label. "AutoSize" doit être faux pour permettre l'utilisation de plusieurs lignes.

- Le composant Bevel a un but simple : il est décoratif. En effet, ce composant ne sert qu'à créer des effets visuels à l'aide de ses quelques propriétés permettant de créer des bordures 3D. Vous pouvez ainsi créer un cadre, ou seulement l'un des côtés du cadre, et changer l'effet graphique du composant. Ce composant, moins performant que Panel dans le sens où il n'est pas un conteneur et offre un peu moins de possibilités de bordures, permet cependant de se passer d'un Panel de temps en temps.

Align — Permet, comme pour les panneaux et d'autres composants, d'aligner le Bevel sur un des bords de son conteneur, voire sur tout le conteneur pour lui adjoindre une bordure si ce dernier n'en possède pas.

Shape — Détermine l'effet visuel créé par le Bevel (dépendant également de la propriété Style :

- bsBox : affiche-le contenu du Bevel comme relevé ou enfoncé.
- bsFrame : affiche un cadre relevé ou enfoncé autour du contenu.

- bsTopLine : affiche uniquement la ligne supérieure du cadre.
- bsBottomLine : affiche uniquement la ligne inférieure du cadre.
- bsLeftLine : affiche uniquement la ligne gauche du cadre.
- bsRightLine : affiche uniquement la ligne droite du cadre.

Style	Fixe le style de bordure du cadre : bsLowered (enfoncé) ou bsRaised (relevé).

X. TIMER :

Le composant de type « Timer » est un composant non-visuel mais doté également de propriétés et de méthodes. Leur nombre est toutefois assez limité.

Les composants de type « timer » sont disponibles dans la palette des composants, dans l'onglet **Système.**

Ils sont munis de deux propriétés importantes :

Enabled : qui indique si le composant est actif ou ne l'est pas.

Interval : qui indique combien de temps doit se passer avant l'activation du seul événement que gère l'objet, **OnTimer**, le temps est indiqué en millisecondes.

APPLICATION

Démarrez une nouvelle application, sur la fiche vierge, déposez :

- Un composant de type **Timer (**la position du composant de type « Timer » est sans importance, on ne le verra pas durant l'exécution du programme, c'est un composant non visuel)
- Un composant de type **Button**
- Un composant de type **Label**
- Affectez la valeur **False** à la propriété Enabled du composant **Timer**, il est donc désactivé.
- Affectez la valeur **3000** à la propriété **Interval.** Il va donc se passer 3000 millisecondes, c-à-d 3 secondes avant le déclenchement de l'événement.
- A l'événement **OnTimer,** affectez la procédure qui permet de passer le composant **Label** en couleur rouge.
- A l'événement **OnClick** du bouton, affectez la procédure qui remet la propriété **Enabled** à la valeur **True.**
- Que se passe-t-il lors de l'exécution de cette application ?

✓ Un clic sur le bouton provoque le démarrage du composant « Timer ».

✓ Lorsque le temps prévu est écoulé, le composant «Timer » provoque l'exécution de la procédure prévue : le changement de couleur du composant Label.

Nous obtenons donc une cascade de deux événements.